Science Arts & Métiers (SAM)
is an open access repository that collects the work of Arts et Métiers Institute of Technology researchers and makes it freely available over the web where possible.

This is an author-deposited version published in: https://sam.ensam.eu
Handle ID: .http://hdl.handle.net/10985/16758

To cite this version :

Harvey ROWSON, Matthieu BRICOGNE, Lionel ROUCOULES, Alexandre DURUPT, Benoit EYNARD - Knowledge capture and reuse through expert's activity monitoring in engineering design - In: PLM Conference, Italie, 2018-07-02 - PLM conference (Torino, Italy) - 2018

Any correspondence concerning this service should be sent to the repository Administrator : scienceouverte@ensam.eu

Knowledge capture and reuse through expert's activity monitoring in engineering design

Harvey ROWSON (1)(3), Matthieu BRICOGNE (1), Lionel ROUCOULES (2)

Alexandre DURUPT (1), Benoit EYNARD (1)

1: Université de Technologie de Compiègne, France
2: Ecole Nationale Supérieure d'Arts et Métiers Aix-en-Provence
3: DeltaCAD Lacroix Saint-Ouen, France

E-mail Address: harvey.rowson@utc.fr

Abstract. This paper deals with artificial intelligence driven product engineering support. Many software systems are available to support the product lifecycle, especially during product design, such as CAD, PDM, CAE, SDM, etc. Most product development process is performed using these systems, which through their rich user interfaces allow skilled professionals to express their expertise and knowledge using the tools and functions the software is willing to provide them. At the end of the day, the result of their work is a model, built through a user interface, and stored in a repository. The goal of our research is to reverse engineer the user's knowledge by analysing his/her actions with the software system, based on the assumption that the process will itself be meta-knowledge driven and that we will focus on engineering software which provide semantically rich user interfaces. The aim of this paper is to investigate the idea of building reusable expert knowledge from actions on engineering software user interfaces. It first outlines existing works from different fields and identifies remaining issues. It then suggest an approach to address these issues and put together an operational system.

Keywords: Artificial Intelligence, Knowledge Based Engineering, Engineering Design, GUI monitoring, Computer Vision

1 Introduction and objective

(Rocca 2012) observe that product engineering could be improved in order to reduce time spent on sometimes repetitive and other times trivial tasks. This would allow designers to focus their attention and time on higher value-added undertakings. They analyse the subject from a Knowledge Based Engineering (KBE) point of view and explores how different paradigms, including Information Technology (IT), Artificial Intelligence (AI) and Computer Aided Design (CAD), can lead to miscellaneous approaches to KBE. One conclusion is that an important improvement would be "lowering the accessibility level", suggesting improving the ability for non-programmers to be able to define the KBE application, in particular by closing the gap between so called KBE languages and natural languages (Rocca 2012).

More generally, (Intharah et al. 2017) state that much time is wasted using software non-creatively, on repetitive and tedious tasks. They introduce an approach based on Computer Vision (CV) and Graphical User Interface (GUI) analysis stated as "learning-by-demonstration in a GUI world". They perform extensive experimentations with these concepts. Although this clearly could help addressing some of the issues observed by (Rocca 2012) a few years earlier, they nevertheless identify a number of shortcomings. These range from misclassification issues of user actions to unawareness of the system's (and software's) state. (Dominic et al. 2016) elaborate an AI based approach to capture, store and reuse knowledge within bridge structure calculation, which is demonstrated through usage of Bayesian networks to optimise parametric design. These works all strive at enhancing the user's efficiency while using software systems, by bringing some assistance in some form, known as KBE in the engineering community.

Our research tackles the problem from a complementary angle, combining many of the previously mentioned approaches, by suggesting the use of what we will call an Engineering Personal Assistant (EPA). An EPA is a software agent designed to help engineers use their dedicated software to perform their product designs. The focus of the EPA is twofold, first it ambitions to capture knowledge by monitoring the usage of the engineering software (it thus learns by example), and second it aspires to reuse this knowledge to help guide the user in future usages (it thus helps by pertinent suggestion). We believe that many by-product applications are possible for both of these aspects, as the captured knowledge could be used in many ways, and the advisory agent could rely on multiple knowledge sources. The EPA concept has some similarities with other forms of assistance to software usage, such as the Microsoft Clippy assistant[1]. This was integrated in office 1997 and abandoned around 2004, it was used mostly as a contextual user friendly online help system. Virtual assistants[2] are other examples, such as Amazon Alexa, Apple Siri, Microsoft Cortana or Google Assistant. These focus mainly on providing a voice control interface to regular actions (dial contact, create calendar event, perform Web search, …).

Our proposal, which is in an early state, aspires to achieve a robust approach, at least conceptually for now, by building upon CV, AI and KBE technologies in a predefined

[1] See https://en.wikipedia.org/wiki/Office_Assistant for additional information
[2] See https://en.wikipedia.org/wiki/Virtual_assistant_(artificial_intelligence) for details

context, which is that of product engineering and design. Regarding the knowledge capture aspect, CV is used to monitor and log the system state and actions. KBE structures and principles, and the focus on product engineering methods and tools, are transverse to the whole approach, providing an overarching domain specific background allowing the necessary paradigmatic foundations. This paper therefore summarises related works from the two main fields which we plan to build upon and then proposes an operational process to enable the vision we have developed, which we plan to implement in a forthcoming demonstrator which is in progress.

2 State of the art of related works

2.1 The field of Knowledge Based Engineering (KBE)

When investigating the state of the art of works targeting a software solution to optimize product engineering activities, KBE is a recurring approach. (Danjou et al. 2008) timestamps the birth of KBE to be around 1980 and present an approach focused on encapsulating knowledge in customized features in order to achieve capitalization and reuse in a CAD based environment, which now underlies many knowledge-based systems embedded in engineering tools, as discussed further.

(Reddy et al. 2015) present a survey of KBE approaches which illustrates the strong activity of this field and the diversity of approaches and results one can observe. They identify KBE as being a paradigmatic evolution over parametric design, with the advantage, amongst others, to be better suited for rapid development of customisable product families. Technologically, they identify KBE to be supported by information technologies, artificial intelligence and knowledge management. The popularity of KBE is rooted in the competitive advantage it brings for efficiently developing new products including customisation and component reusability. With figure 1, they clarify the concept of knowledge engineering and propose a diagram illustrating how it relates to its environment.

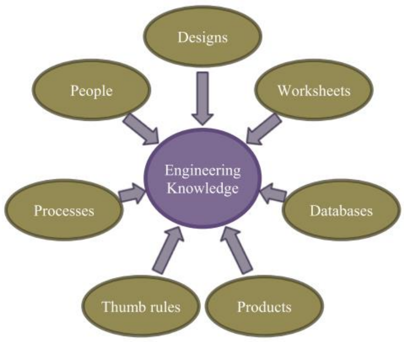

Figure 1:Different elements of knowledge according to (Reddy et al. 2015)

According to their analysis, 80% of design activities can be considered to be recurrent, and KBE can have great impact on these activities. An important step identified for KBE implementation is that of knowledge capturing and formalisation, and they list a number of methods and technics targeting this activity, such as MOKA, KOMPRESSA, DEKLARE or DEE (Reddy et al. 2015) whisle noticing that these produce knowledge bases which are not readily usable, as they require some tooling technology to support them. On the other hand, they also observe that engineering tool providers also implement internal knowledge-based systems [3]. They however state that this produces somewhat limited knowledge basis, as they tend to be specific to the system rather than the field, and also the knowledge is not easily extractable and reusable in different environments. They finally conclude that one of the strongest drawbacks of current KBE approaches, regardless of the two previously mentioned paths, is the difficulty to collect and formalise the knowledge in a robust manner, and believe better usage of Wiki like methods could bring solutions to this.

(Quintana-Amate et al. 2015) take a focused look at the knowledge sourcing issue. They first account for the successful usage of KBE systems in many industries, with more in-depth analysis of an Aeronautic example, and some of the issues encountered. One of these is the recurring difficulties raised by the need to fuel the KBE system with the required knowledge in the required formalism (often parameterized rules and constraints). They propose an extensive review of methods targeting this upstream phase of most KBE projects, and conclude that this area requires more work in order to solve the many remaining issues and to improve the overall process. They follow with a review of different related approaches in order to suggest finally a set of technologies which are identified as being of higher potential to address the knowledge generation phase. They suggest that artificial intelligence approaches should be pushed to handle this capitalization phase in order to significantly reduce the need of manual structuration and input of information to generate the required knowledge. They conclude with the concept of "learning by doing", which should be promoted in order to use artificial intelligence in a partially automated knowledge acquisition process.

Focusing on civil engineering, and more specifically bridge structure optimization, (Dominic et al. 2016) use modern artificial intelligence results, in the form of Bayesian networks, to establish a model and a method which is able to learn optimization strategies through analysis of existing studies within a bridge management system. They use a training set to perform machine learning processes which result in a fitting of the Bayesian network they designed. Although they do not demonstrate the approach's ability to support advisory actions, they believe this is possible for example by using machine learning technics to bridge the remaining gaps in the different modules of their knowledge management system. In (Dekhtiar et al. 2018), a review on the use of deep learning in CAD and PLM is also proposed.

Section §3 analysis these elements in regards of the EPA objectives and outlines the main aspects which our proposal (in §4) should focus on in order to fulfil its objectives, in particular automating the capitalisation phases and building a transparent while pervasive solution.

After this look into the field of product engineering, we will now turn towards the field of software testing in which a wealth of highly valuable input can be taken from. As illustrated in the following section, one approach to software testing is the technic consisting in capturing and replaying action sequences on the software through its GUI (in a similar way a human tester would do it).

---

[3] Some examples of this approach could be Unigraphics NX Knowledge Fusion or Dassault Systemes CATIA Knowledgeware

2.2 The field of Software Testing and Automation

Software editors, since the advent of GUIs, have faced the daunting task of testing and validating their products. Batch and script-based testing of command file applications have long driven the field of software testing, but these approaches fall short when it comes to software which is not only driven by such command files, and which results are not only so-called output files. The process for testing this type of software, without GUI, is basically to create a test case suite, with a set of input command files and a set of corresponding expected output files. These are then run to check that new versions of the software produce acceptable results as compared to the reference output files.

Although this does present some challenges of its own (especially when the notion of acceptable is fuzzy), it certainly misses the point of testing the GUI layers. (Alégroth and Feldt 2017) seek new solutions to this problem, and start by establishing a clean taxonomy of the existing approaches they have collected from existing tools and existing academic works, focusing on validating the GUI layers of software systems. They identify three distinctive steps in the history of the art of software testing. They go on to classify following three categories: pixel coordinate approaches, event interception methods and computer vision paradigms.

(Qureshi and Nadeem 2013) propose an extensive analysis of approaches falling into the second generation identified by (Alégroth and Feldt 2017). The interesting point of this is that it illustrates the maturity of this field in contrast to the third generation which is still emerging at the time. Their work breaks down the second generation into 12 families of methods, synthesized in a table classifying them following a number of interesting criteria, namely: Input representation of GUI under test, Intermediate representation, Coverage criteria, Automation, Tool Support, Case study, Fault model, Fault injection. This is illustrated in the table 1.

		Evaluation Parameters		
		Input representation of GUI under Test	*Intermediate representation*	*Coverage Criteria*
1.	R. Shehady *et. al* (1997)	Input, Output, Variables,States	VFSM Model, FSM Model which is used by Wp Method	All-paths, All-transitions
2.	A.M.Memon *et. al.* (99, 00, 01)	Source file of GUI	Hierarchical Model Scenario (Initial State, Goal State, Operators, Objects)	All-transitions
3.	S. R. Dalal *et. al.* (1999)	Req. & constraints of GUI	Data Model (Specification of inputs for AETGSpec)	Pairwise-interaction event-interaction
4.	L. White and H. Almezen (2000)	Identification of CIS	Reduced FSM (Trace all distinct paths)	All-paths, event-interaction
5.	Belli (2001)	Identification of CIS & FCIS	FSM Model (Trace all valid & invalid paths)	All-paths, event-interaction All-IPs, All-FIPs
6.	Kai-Yuan Cai *et.*	Identification of	Mealy machine/ FSM Model(Convert	All-path,

Table 1:Methods from (Qureshi and Nadeem 2013)

(Börjesson and Feldt 2012) state the importance of focusing on the GUI layer of the system under test (the third generation of approaches reported by (Alégroth and Feldt 2017)). They first proceed to outline and analyse the weaknesses associated to event capturing and replay, as done within second generation approaches and summarized above. They then detail existing methods relying on Visual GUI Testing approaches, and provide further analysis of two tools, one which is a commercial software system and the second is the open source software application. They demonstrate the usage of these systems on some experimental cases and conclude to some of the limitations and difficulties mentioned above. They also point out some interesting distinctions between the fields of software testing and software automation, as they point out that the first have a focus on exhaustivity and coverage whereas the second have a focus on understanding and intelligence.

(Moreira and Lopes de Matos 2014) provide some important complementary ideas resulting from their work within the European "FCOMP-01-0124-FEDER-020554" project. Their approach strives to integrate and leverage the fact that GUIs often are built and designed relying on partly shared and standard design patterns and schemes (linked, amongst others, to ergonomic and implementation considerations). They propose a specific model, named DSL PARADIGM, which is able to model these similarities and shared features, and they demonstrate its usage on a number of test cases.

These elements are further analysed in §3 in the light of the EPA's objectives in order to identify the main aspects our approach (described in §4) should focus on, particularly targeting a specific type of software and activity, leveraging use interface richness and accepting extensive customisation.

The existing works summarised here within these two areas, although non-exhaustive, show that many scaffolding principles of our approach (detailed in §4) are already quite developed and should be built upon to elaborate the conceptual and technical elements of our solution. Overall, the field is wealthy and attracts high interest from the international product engineering and software engineering communities, with no sign of slowing down, and with new paradigms regularly emerging and yielding new hopes (for example around the current artificial intelligence trend). It also shows a number of areas where things could be improved if the idea is to promulgate higher robustness and efficiency of product engineering activities, as presented in the below section.

3 Towards a new kind of software to enhance user support and efficiency

This section analysis the above state of art at the light of our research objectives, which, as stated in introduction, is to propose an EPA whose purpose is to support efficient knowledge capitalisation and reuse within product engineering activities. More precisely, it aims at capturing the expert's knowledge by monitoring the expert's interactions with the engineering software she/he is using, in a somewhat similar way a human could learn from observing someone else perform a task. However, here we limit the span to an activity consisting in using a software system and we only consider the explicit interactions between the user and system. We expect this capitalisation phase to rely on computer vision (CV) to capture the on-screen actions, and to use some form of advanced processing to transform these actions into reusable knowledge. The EPA then aims at reusing this knowledge through proposals to the user while she/he is interacting with his software. This supposes an identification of the ongoing interaction as being related to a capitalised interaction (again, likely relying on CV and advanced processing). It also implies being able to fit the capitalised interactions to any existing variations, and only then suggesting some support to the user. This can be for example in the form of automating the end of a multi-step but recurrent interaction.

The objective therefore leads us to identify a number of specificities relative to the current state of art in the identified fields we summarised in §2.

In particular, relative to the field of software testing, it can first be observed that the EPA focuses on a specific type of software, which supports a specific type of activity, respectively engineering support software and product engineering activities. The assumption is that specialising the concepts to this more specific field will reduce complexity and help achieve robust results in capturing interactions. Second, an interesting characteristic of the targeted software (CAD, CAE and PLM systems) is that they display rich user interfaces. This means they have a large variety of widgets (buttons, lists, checkboxes, fields, toolbars, menus, …) and a structure which is viewed as a facilitator in making sense out of the actions. They are hence coined as being semantically rich user interfaces. Third, the method integrates the idea of customisation, meaning that initial work is accepted to adapt the solution to any specificity identified in the target environment which seems of help to achieve the goal. This opens the path to parametrising and fine tuning the approach as to account for variabilities which could otherwise tend to hinder robustness. In particular, this aspect allows to establish a meta-knowledge base (which can include any information deemed useful for the purpose, including in particular static data, dynamic data, algorithms and workflows) which we can use to guide our overall process, more of which will be said in §4.1

Relative to the field of KBE, one can observe first that a valuable aspect here is the automating of the capitalisation phase through automatic observation and learning. This is seen as complementary to widespread knowledge capitalisation methods mentioned above in §2.1. Second, the EPA, as its name implies, is a software agent which should be as transparent as possible to the user, who should barely know of its existence except from benefitting from its suggestions and proposed automations. The EPA should seamlessly blend into the user's environment (operating system and engineering software) and will not replace any other system and will not bring any constraints (only possibilities). Third, the solution should not be hard linked to any specific software, and in particular, it should not require changes to the existing software, unlike for example KBE solutions which could be integrated into the CAD systems (Workbenches in CATIA for example). Fourth, the two aspects of the EPA, namely knowledge capturing and knowledge reuse, could be leveraged independently, assuming that the captures knowledge could be of interest for many applications and the advisory engine could maybe be powered by knowledge from other sources than its own capturing.

The research proposal, which is currently in early state and is planned to be developed during upcoming works, is presented in the following as we understand it currently. It consists mainly in building the EPA system as characterised above, starting by outlining its main architectural dimensions and its major operational principles. Different methods are planned for the validation of its conceptual and operational capability. A first goal is to use the EPA to monitor usage videos (such as Catia tutorials found on youtube) and extract and capitalise knowledge from these. A second will be to reuse the acquired knowledge in order to replay the tutorials progressively different settings, for example by first changing technical aspects (screen resolution, colour depth, …) and going towards higher level changes (initial model state, adaptable parameters, …).

4 How to help engineers use product engineering software

The envisioned approach is hence based on the elements introduced above with the main idea being to bridge the gap between existing concepts and the target goals. This section brings further details around these and provides an overall overview of the conceptual framework that is planned to be experimented in future works, as for now it is more of a scaffolding than a framework. To ease understanding, the solution is presented as if modules were independent, processes were sequential and things are monomorphic (viewed along one perspective), more of which will be said at the end of this section. We will first look at the capitalisation process, which will bring us also to the concept of meta-knowledge. We will then move onto the reuse phase, and elaborate on potential by-products the system can open to.

4.1 Capitalisation through monitoring and meta-knowledge

A central aspect of the solution is that of monitoring the user's interactions. Monitoring in this context is the activity of capturing and tracing the interactions the user has with the software she/he is using (Satama 2006). Examples of such interactions could be when the user activates a button, enters values into fields, activates a toolbar, selects a menu, … (Sadeghi et al. 2016) present a model of such a process, where they relate the development of a product part with the sequence of actions on a CAD software and the related impact within the product model. With this first step in mind, the next step is to look at how this can be performed. As presented in §2, work has already been done on capturing user actions on graphical user interfaces, and many results have been achieved within these works, whether to bring automation or to support software testing activities. As the EPA is non-intrusive, any approach that involves instrumenting of the target software will be discarded. Therefore, screen content analysis will be the preferred path, using mainly computer vision technology applied to identifying GUI widgets on the screen, and the changes of state of these to help identifying actions without needing to hook (Memon et al. 2003) the operating system's event stack.

To simplify matters, and with no apparent drawback in the frame of our work, it is planned to help this process by providing it with as rich and extensive as necessary data beforehand. (Satama 2006) explore this via the idea of domain specific models to ease testing. This will be part of the aforementioned meta-knowledge base which will be elaborated as required and with accepted overhead work. In particular, this will include information about the possible graphical representations of the interaction widgets (buttons, toolbars, …) and the nature of the actions these widgets generate. This includes contextual variations and the means to identify them as such. For example, the fact that a same widget could perform different actions depending on how it's embedded in the GUI hierarchy. This should be achievable by combining the computer vision technics presented, such as those overviewed in (Leo et al. 2017), and possibly pushing further towards machine learning algorithms in order to benefit from multi representation of each widget, and the purposely created widget base which will be designed specifically for the targeted scenarios.

4.2 Reusing through contextual similarity identification during operations

The primary usage of this knowledge base within the EPA paradigm is its reuse to help the user perform tasks where the capitalised knowledge can help automate sequences of tasks the user is encountering. The first step to make this possible (assuming the knowledge base exists that is) is to identify in the current flow of actions an opportunity for reuse. This means analysing the ongoing interactions, with the same technologies and methods as during the capitalisation phase, and identifying in real time a resemblance between the script under elaboration and schemes in the knowledge base. Obviously, the difficulty is predicting the extrapolation of the script, as the script is under construction while the scheme is completed, and the goal is to estimate the chances of any given ongoing script to result into an existing scheme. If a match is identified, and an existing scheme is deemed pertinent regarding the current interactions, the EPA can suggest help to the user. It should present the proposed scheme, and the potentially required customisations. The EPA can then finalise the interaction sequence for him in an automatic manner using again computer vision technologies to identify related widgets and activate them as in (Chang et al. 2010).

It should be noted here, that the usage process is in fact possible regardless of the capitalisation process, providing that the EPA has access to the knowledge in the required formalism, which for testing purposes could be hand produced. This is an interesting feature as it allows working on the subject in a breadth first method rather than a depth first, the latter been subject to deadlock should the capitalisation process reveal unexpected shortcomings. Symmetrically, the capitalisation process can hold value even if the reuse process falls short. Indeed, analysing the usage logs, scripts and schemes could be used in many ways, for example to help promote and enhance best practices or standards across users and teams. Both the capitalisation and reuse aspects can also be imagined coupled with existing KBE systems, either as tools to generate the knowledge, or as agents to use it on the fly, providing that interoperability concerns are accounted for in the knowledge structures and operational principles.

5 Conclusions

This paper has presented a global vision of the Engineering Personal Assistant (EPA) as a Knowledge Based Engineering (KBE) inspired approach hoping to help product engineering and design activities by bringing partial automation of software usage. However, it takes a complementary approach to the current ones, by focusing on automatic capture and learning of the knowledge from monitoring of the user's interactions on the software. State of the art artificial intelligence technologies are the foundational enablers of the process, particularly computer vision linked to machine learning and template matching, and text processing of the logs to extract and build knowledge. Lots of work is planned in putting all this together and elaborating the central informational structures. Current work is focused on formalising the concepts (and models) while experimenting the envisaged technologies, with already some issues identified with the computer vision aspects for example (which seem more sensitive to video encoding and artefacts than expected), but workarounds seem possible.

6 References

Alégroth E, Feldt R (2017) On the long-term use of visual gui testing in industrial practice: a case study. Empir Softw Eng. doi: 10.1007/s10664-016-9497-6

Börjesson E, Feldt R (2012) Automated system testing using visual GUI testing tools: A comparative study in industry. Proc - IEEE 5th Int Conf Softw Testing, Verif Validation, ICST 2012 350–359. doi: 10.1109/ICST.2012.115

Chang TH, Yeh T, Miller RC (2010) GUI testing using computer vision. Proc 28th Int Conf Hum factors Comput Syst 1535–1544. doi: http://doi.acm.org/10.1145/1753326.1753555

Danjou S, Lupa N, Koehler P (2008) Approach for Automated Product Modeling Using Knowledge-Based Design Features.

Dekhtiar J, Durupt A, Bricogne M, Eynard B, Rowson H, Kiritsis D (2018) Deep learning for big data applications in CAD and PLM – Research review, opportunities and case study. Comput Ind 100:227–243. doi: 10.1016/j.compind.2018.04.005

Dominic S, Bügler M, Borrmann A (2016) Knowledge based Bridge Engineering - Artificial Intelligence meets Building Information Modeling.

Intharah T, Turmukhambetov D, Brostow GJ (2017) Help, It Looks Confusing. In: Proceedings of the 22nd International Conference on Intelligent User Interfaces - IUI '17. ACM Press, New York, New York, USA, pp 233–243

Leo M, Medioni G, Trivedi M, Kanade T, Farinella GM (2017) Computer vision for assistive technologies. Comput Vis Image Underst 154:1–15. doi: 10.1016/j.cviu.2016.09.001

Memon A, Banerjee I, Nagarajan A (2003) GUI Ripping: Reverse Engineering of Graphical User Interfaces for Testing.

Moreira R, Lopes de Matos M (2014) Pattern-Based GUI Testing. PQDT - Glob 180.

Quintana-Amate S, Bermell-Garcia P, Tiwari A (2015) Transforming expertise into Knowledge-Based Engineering tools: A survey of knowledge sourcing in the context of engineering design. Knowledge-Based Syst 84:89–97. doi: 10.1016/j.knosys.2015.04.002

Qureshi IA, Nadeem A (2013) GUI Testing Techniques: A Survey. Int J Futur Comput Commun 142–146. doi: 10.7763/IJFCC.2013.V2.139

Reddy EJ, Sridhar CN V., Rangadu VP (2015) Knowledge Based Engineering: Notion, Approaches and Future Trends. Am J Intell Syst 5:1–17. doi: 10.5923/j.ajis.20150501.01

Rocca G La (2012) Knowledge based engineering: Between AI and CAD. Review of a language based technology to support engineering design. Adv Eng Informatics 26:159–179. doi: 10.1016/j.aei.2012.02.002

Sadeghi S, Dargon T, Rivest L, Pernot J-P (2016) Capturing and analysing how designers use CAD software.

Satama M (2006) Event Capturing Tool for Model-Based GUI Test Automation.